In the Claims:

1. (Currently Amended) A method for at least semi-automatically translating code written in a first language to a second target language, ~~the first language featuring structural constraints and featuring dynamic behavior, wherein the first language features a hierarchy of objects, such that relationships between the objects are determined by constraints,~~ the method comprising:

detecting an underlying control structure for the code in the first language, wherein the first language features structural constraints, dynamic behavior, and at least one hierarchy of objects, respective ones of said objects in said hierarchy having relationships, said relationships between the objects being determined for said code by said structural constraints, and said language imparting said dynamic behavior to said code, said underlying control structure incorporating said relationships and said dynamic behavior;

creating a static framework of resources for supporting the dynamic behavior ~~of~~ imparted to said~~the~~ code written in the first language; and

mapping the dynamic behavior to the second target language according to said static framework of resources and said underlying control structure.

2. (Original) The method of claim 1, wherein said underlying control structure is detected by control flow analysis for an action, to determine at least one of a condition and a trigger for causing said action to execute.

3. (Original) The method of claim 2, wherein said at least one of a condition and a trigger is a guard for governing execution of said action, such that control flow analysis comprises at least determining at least one guard for each action.

4. (Original) The method of claim 3, wherein said control flow analysis further comprises determining a plurality of guards for creating a sequential control flow graph.

5. (Original) The method of claim 4, wherein said underlying control structure for the code in the first language is detected by:

parsing the code; and

creating an abstract syntax tree;

wherein said at least one node of said sequential control graph retains a reference to a node of said abstract syntax tree.

6. (Previously Presented) The method of claim 4, wherein said control flow analysis comprises:

computing a triggering structure for each process of the code in the first language;

determining said at least one guard for each action of said process; and

determining a segmentation of said process into a plurality of segments.

7. (Original) The method of claim 6, wherein said control flow analysis further comprises:

unrolling at least one loop.

8. (Original) The method of claim 6, further comprising retiming at least one action.

9. (Original) The method of claim 6, wherein each node is selected from the group consisting of a basic node for representing a list of non-time consuming actions, a guard node for representing a branch point, and a wait node for containing a temporal expression.

10. (Original) The method of claim 2, wherein said control flow analysis detects at least one malformed control structure for manual alteration by a user.

11. (Original) The method of claim 1, wherein said static framework of resources is created by elaboration, wherein elaboration is performed by allocating a sufficient number of state holding elements for representing dynamic behavior of the code written in the first language.

12. (Original) The method of claim 11, wherein said state holding elements are allocated by:

recursively analyzing a structure of the code written in the first language;

determining structural constraints; and

creating an elaboration graph from said structure of the code and said structural constraints.

13. (Original) The method of claim 12, wherein said elaboration graph features a plurality of nodes selected from the group consisting of scalar nodes, struct nodes and list nodes.

14. (Original) The method of claim 12, wherein said elaboration graph is unfolded to completely represent a plurality of structures of the first language.

15. (Previously Presented) The method of claim 14, wherein the first language features symmetry, and wherein said elaboration graph is unfolded to overcome an asymmetrical feature of the code.

16. (Original) The method of claim 15, wherein said asymmetrical feature is selected from the group consisting of a temporal expression and a time consuming method.

17. (Original) The method of claim 12, wherein said underlying control structure is detected by control flow analysis for an action, to determine at least one of a condition and a trigger for causing said action to execute, wherein said at least one of a condition and a trigger is a guard for governing execution of said action, and wherein a plurality of guards is determined for creating a sequential control graph, the method further comprising:

determining an interrelationship between said elaboration graph and said sequential control graph.

18. (Original) The method of claim 17, wherein said interrelationship is used to detect a race.

19. (Original) The method of claim 17, wherein said interrelationship is used to compute an execution schedule.

20. (Original) The method of claim 1, wherein the first language features at least one type of symmetry, and wherein said at least one type of symmetry is maintained when creating said static framework of resources.

21. (Original) The method of claim 1, wherein the first language is a verification language.

22. (Original) The method of claim 1, wherein the code, after translation to the second language, performs verification of a design.